



118, route de Narbonne
31077 Toulouse Cedex 4

Développement d'une carte à processeur linux



Rapport de TER
M1 SME

Auteurs : KERVELLA Guillaume, BENAKCHA Samir
Intervenant : BERTHOU Pascal
Master1 SMI

Promotion : 2013-2014

Introduction

Dans le cadre du Master 1 SME, nous avons eu l'opportunité de réaliser un Projet TER (travaux d'études et de recherche) dans le but de découvrir des nouvelles technologies dont nous devons par le biais de la recherche, assimiler le fonctionnement pour ensuite se lancer dans la réalisation proprement dites. Ce projet d'étude et de recherche est effectué en binôme, afin d'être aiguillé dans notre étude, l'enseignant qui propose le sujet est régulièrement en contact avec le binôme. Le projet est non seulement un travail théorique mais celui-ci devra comporter en plus d'une partie simulation.
Dans un premier temps, nous avons dû se mettre en binôme après avoir pris connaissances des différents sujets proposés par les enseignements, pour ensuite choisir un projet qui ce déroulera sous la responsabilité de M. Pascal Berthou.
L'intitulé de notre projet est le suivant :

- Manipulation des E/S de la carte et développement d'exemples très simples d'utilisation.
- Cross-compilation et Installation de Linux sur la carte. Rédaction d'un cook-book à destination des futurs étudiants.

Cette carte de développement sera l'objet d'études pour les futurs étudiants, cette étude se présentera sous forme de travaux pratiques. Afin de réaliser ces travaux pratiques les étudiants devront avoir à leurs dispositions un PC équipé d'un noyau linux, d'un réseau FTP ainsi que d'un cook-book dont nous avons la responsabilité de concevoir.
Durant notre projet, il nous a été confié la responsabilité de faire l'étude, la conception, la réalisation de programme permettant la mise en œuvre des différentes entrées et sorties de notre carte.
Pour l'étude de la carte, il a été nécessaire au préalable de se familiariser avec les systèmes embarqués sous Linux. Pour cela nous avions réalisé une recherche bibliographique, qui par la suite a contribué à la création d'un état de l'art au sujet des systèmes embarqués.
Un noyau Linux a déjà été installé sur la carte de développement. Pour rendre la communication possible avec notre carte, nous avons procédé à l'installation d'une chaine d'outils contribuant au développement de celle-ci. Pour construire tous ces outils, il a fallu un système de construction et pour cela nous avons utilisé « Buildroot ».
N'ayant jamais utilisé ce système de construction au cours de notre formation, nous avons d'abord fait un apprentissage afin d'utiliser au mieux les différentes fonctionnalités proposées par cet outil. Buildroot nous a permis par la suite de construire une chaine de compilation croisée GCC afin de générer sur le PC des logiciels qui pourront être exécutés sur la carte.
Ensuite nous avons conçues des programmes simples, mettant en œuvre les différentes entrées et sorties de la carte. Ces programmes ont été conçus à partir de fonctions et bibliothèques fourni par le constructeur Armadeus. Quand les résultats obtenus lors des différentes simulations ont été jugés satisfaisantes, il est décidé de valider le programme.

Bibliographie
Les systèmes embarqués :

Histoire :

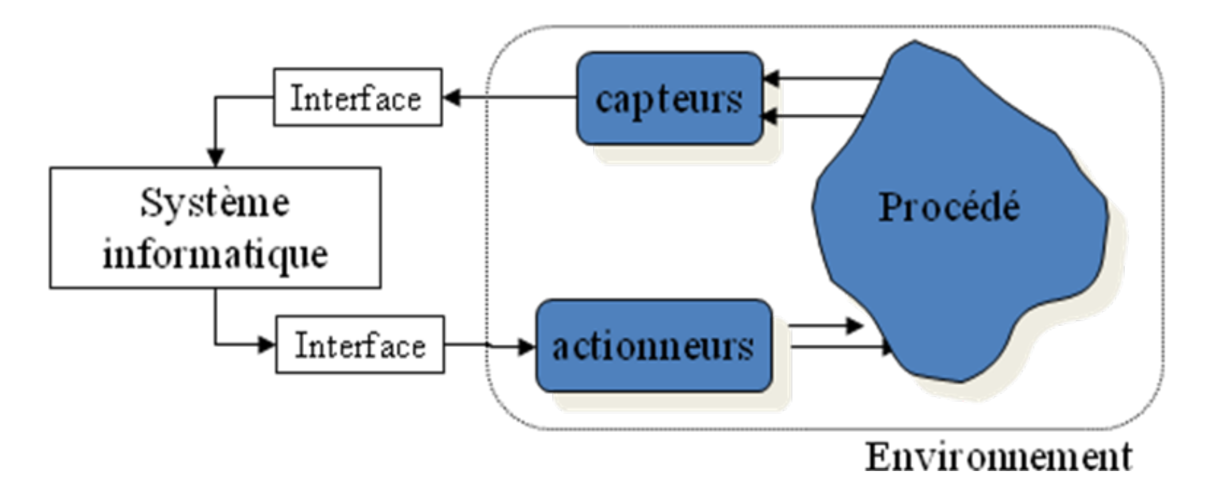
Les premiers systèmes embarqués sont apparus en 1971 avec l'arrivée sur le marché de l'Intel 4004 [1]. L'Intel 4004 est le premier microprocesseur, incorporant toutes les parties d'un ordinateur dans un seul et même coffret, rassemblant : unité de calcul (UAL), mémoire, contrôle des entrées et sorties [2].
Avant la création des microprocesseurs, il fallait plusieurs circuits intégrés différents, chacun dédié à une tâche particulière, alors qu'aujourd'hui un seul microprocesseur peut assurer autant de tâches différentes que sa mémoire lui permet. Ceci a été une innovation majeure, puisque des objets du quotidiens tels que fours à micro-ondes, télévisions ne tardèrent pas à être équipés de microprocesseurs.
Le premier système moderne embarqué reconnaissable a été le « *Apollo Guidance Computer* », le système de guidage de la mission lunaire « Apollo », développé par Charles Stark Draper du Massachusetts Institute of Technology [3].
Au début du projet, le PC d'Apollo était considéré comme l'élément le moins robuste du projet. Par contre grâce à l'utilisation de nouveaux composants qu'étaient à l'époque les circuits intégrés, des gains particulièrement importants sur la place utile et la charge utile ont été accomplies, avec une diminution conséquente des risques déjà nombreux lors des missions [4].
La demande des systèmes embarqués est souvent beaucoup plus importante que les ordinateurs de bureau. En 1999, il s'est vendu 1.4 milliard de processeurs 8 bits pour systèmes embarqués contre 104 millions de processeurs pour PC. Pour limiter les coûts on contraint donc la quantité de mémoire disponible, les fonctionnalités matérielles et la consommation d'énergie.

Définition :

Un Système embarqué est un système électronique ou informatique conçu pour réaliser une ou plusieurs tâches précises. Quelle que soit la nature et la complexité du système, on décompose un système embarqué en 2 parties :

- le système contrôlé
- le système de contrôle

Le système contrôlé est un environnement équipé d'une batterie d'instrument qui réalise l'interactivité avec le système de contrôle.
Le système de contrôle est constitué par des éléments matériels tels qu'un microcontrôleur et logiciels dont le but est d'agir sur le procédé à travers les actionneurs, en fonction de l'état de ce procédé indiqué par les capteurs de manière à maintenir ou conduire le procédé dans un état donné.



Un système électronique embarqué est un élément qui est inclus dans un système plus complexe pour lequel il rend des services bien précis. Celui-ci est constitué de parties matérielles et logicielles qui sont conçues spécialement pour réaliser une fonction dédiée [5].

Description des systèmes embarqués :

On parlera d'un système embarqué lorsqu'un ensemble de logiciels et de matériels est conçu pour une application précise contrairement à un système ordinaire qui peut effectuer toutes sortes de tâches tel qu'un PC. On parle alors de codesign où le matériel et le logiciel sont parfaitement adaptés à la tâche pour lequel le système embarqué est destiné. Exemples de systèmes embarqués :

- Ordinateurs de bord d'une automobile, d'un avion, d'une navette spatiale ;
- Radars, les sonars, les satellites ;
- Smartphones, les routeurs ;
- Robots, les automates programmables, les contrôleurs d'usine, de périphériques industriels ;
- Appareils médicaux ;
- Systèmes d'alarmes, les contrôleurs de climatisation, les ascenseurs ;
- les distributeurs automatiques, les télévisions, les photocopieurs, les caméscopes ;
- les autos radio, calculateur d'airbag, distributeur de boissons, téléphone mobile, console de jeux...

Les systèmes embarqués utilisent généralement des microcontrôleurs pas nécessairement très puissants mais bien adaptés à la tâche demandée. Généralement, le temps d'exécution de la tâche doit être connu de façon qu'elle soit bornée et le système doit être fiable et robuste. Les systèmes embarqués sont très souvent des systèmes temps réel.

Un système temps réel est un système qui interagit avec un environnement externe qui lui-même évolue dans le temps. Le comportement correct d'un système temps réel dépend, des résultats attendus, mais aussi du temps auquel les résultats sont produits [6].

Les différentes technologies de systèmes embarqués :

Microcontrôleurs :

Le microcontrôleur correspond au cœur du système embarqué. Il exécute les instructions embarquées dans la mémoire qui va contenir les instructions du programme pilotant l'application à laquelle le microcontrôleur est dédié. Il gère les ports d'entrées/sorties, les convertisseurs A/N, les Timers, et dispose de mémoires programmable ou non [7].
Le microprocesseur est une parie du microcontrôleur qui prend en charge la partie traitement des informations et envoie des ordres. C'est donc lui qui va exécuter le programme embarqué dans le microcontrôleur.

Avantages :
- Très bon marché
- Apprentissage rapide
- Développement rapide

Défauts :
- Structure fixe
- Extensibilité difficile
- Code non réutilisable

FPGA:

Un FPGA est un circuit reprogrammable, il est constitué de blocs logiques préconstruits et de ressources de routage programmables. Autrement dit, les FPGA sont totalement reconfigurables il suffit de charger une nouvelle configuration de circuits.

Avantages :
- Bon marché
- Architecture flexible
- Ip sur mesure

Défauts :
- Peu intelligents

Processeurs softcore:

Un processeur softcore est un processeur implémenté sur un système reprogrammable comme un FPGA. On parle alors de système sur puce programmable.

Avantages :
- Choix du processeur
- Une « puce »
- Portable

Défauts :
- Peu performant

Linux

Linux est un système d'exploitation libre de type UNIX lancé par le finlandais Linus Torvalds en 1991. Développé sous licence GPL, ce qui donne une disponibilité des codes source gratuitement, a permis d'avoir l'aide au développement du système de développeur dans le monde et a fait le succès de ce système d'exploitation.[8]

Linux embarqué

Définition :

Linux embarqué est un système d'exploitation basé sur Linux et adapté aux systèmes embarqués. Contrairement aux systèmes Linux classiques, Linux embarqué est conçu pour des systèmes limités en ressources. Il utilise généralement peu de RAM et favorise l'utilisation de mémoire flash plutôt que de disques durs. [8]

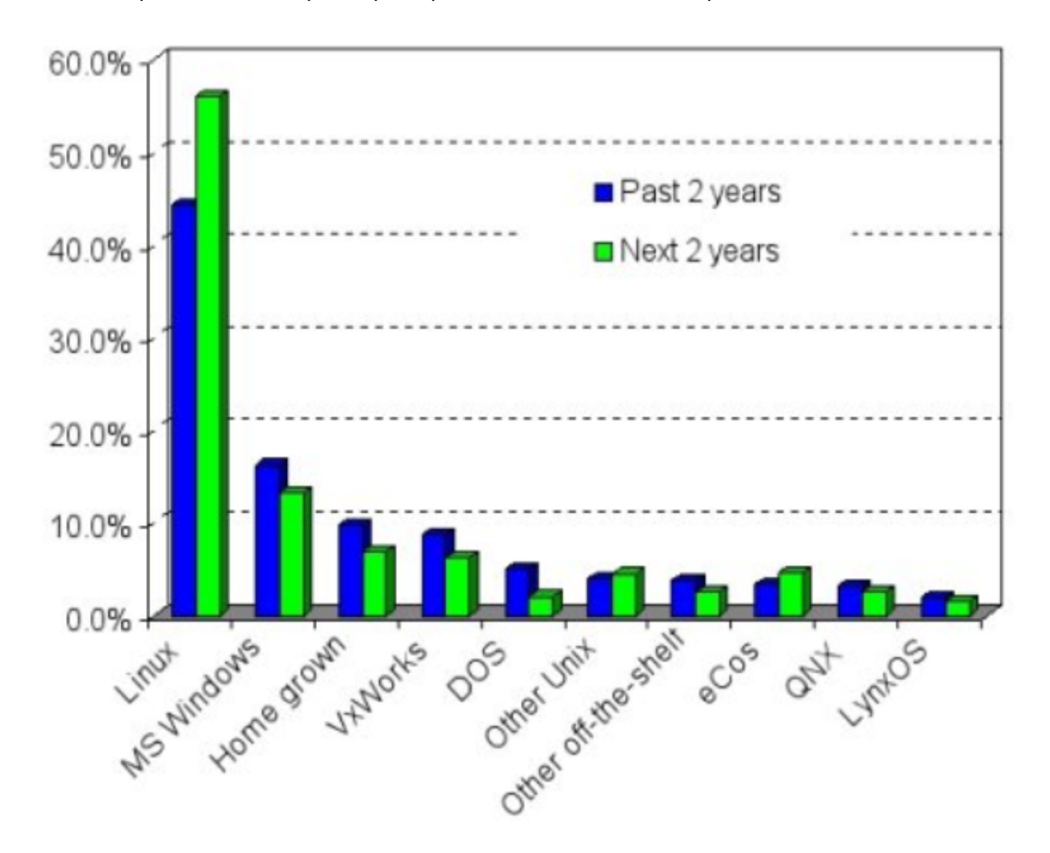
Historique:

1969 : Ken Thompson des laboratoires Bell écrit un système d'exploitation sur un mini ordinateur PDP7, en langage assembleur. Proposé par Brian Khernighan ils nommèrent le système UNICS, puis plus tard UNIX.
1971 : Ecriture de UNIX en langage C afin de facilité sa portabilité.
A partir de 1975: Distribution de UNIX dans les universités.
1979: UNIX atteint sa version 7
Fin des année 1980: Deux versions principales de UNIX cohabitent.
1990: Le langage C est normalisé par l'ANSI (American National Standards Institute) et l'ISO (International Standards Organisation).
1991:Linus Torvalds développe un système d'exploitation sur les bases de UNIX. Epoque d'un internet naissant, son projet suscite beaucoup d'intérêt dans le monde entier et les contributions affluent, notamment grâce au libre accès au code source (déposé sous licence publique GNU).
1992: Le projet nommé LINUX, a une version considéré comme pleinement opérationnel
1996: Linux est utilisé en production industrielle.
2001: Début des projet embarqué avec Linux et sa version 2.4 dédié à l'embarqué.
2008 : les version 2.4 et 2.6 se rencontre équitablement dans les systèmes.
2011: majorité de projet en version 2.6, on commence a parler d'une version 3.0.

Domaine d'application :

Linux embarqué touche divers segment de marché. Les marchés les plus anciens sont celui de la télécommunication et le militaire, sous divers aspects. Mais le marché le plus prometteur et en très forte croissance est l'électronique grand publique. On retrouve le système d'exploitation dans les routeurs, modems, lecteurs DVD/DivX, les disques durs multimédia, la téléphonie (via l'avènement des Smartphones). Mais aussi dans l'automobile avec les divers services que peuvent fournir l'électronique dans un véhicule. Les projets utilisant Linux sont très diversifié et parfois là où on ne les attend pas.


Linux embarqué devient la principale plateforme de l'embarqué



Cross-Compiling:

Lorsque l'on veut installer un Système d'exploitation sur notre cible(système embarqué, carte, etc ...), il se peut que notre ordinateur de développement et notre système embarqué n'aient pas les même architectures, et ceci entrainera des problèmes lors des compilations. Pour remédier à ce problème on utilise la compilation croisée, ce qui consiste à la construction d'exécutables pour une architecture cible, différente de l'architecture sur laquelle fonctionne le compilateur. Les exécutables créés ne pourront qu'être exécuté sur le système cible et non sur la plateforme de développement. [9]

APF28 de Armadeus Systems:

Armadeus Systems

Armadeus systems développe et produit des **systèmes embarqués Linux** (open source) , faible coût alliant petite taille, basse consommation et connectivité étendue.
Ils développent aussi l'association Armadeus Project qui a pour but d'aider toute personne voulant développer des systèmes Linux embarqués libres.

Description de la carte

L'APF28 est une carte à microprocesseur de taille réduite qui ne dispose pas de FPGA.



Caractéristiques de la carte

La carte est équité de:
un microprocesseur i.MX28x (ARM9) à 454MHz,
128 Mo RAM DDR2,
 256 Mo de FLASH SLC NAND,
un ou deux port Ethernet 10/100Mbits,
2 CAN,
un port USB 2.0 High Speed et d'un USB OTG,
Elle est facilement intégrable dans un système embarqué grâce notamment à ses régulateurs et ses convertisseurs de niveau (USB/Ethernet). [10]

Outils nécessaires



Buildroot :

Buildroot est un logiciel qui permet de générer des systèmes linux embarqués complets. Nous pouvons utiliser facilement la compilation croisée et créer le système de fichier racine nécessaire à Linux embarqué. Buildroot est surtout utile pour les personnes travaillant avec des systèmes embarqués , en utilisant différentes architectures de processeur . [11]

Kermit:

Kermit est un Protocol de transfert et de gestion de fichiers. Dans notre étude Kermit nous servira a nous connecter et a contrôler la carte ARMADEUS APF28.

Serveur FTP:

Pour envoyer des fichiers sur notre carte embarquée, on ne peux pas utiliser le protocol Kermit. Pour transférer des fichiers nous devons utiliser le port Ethernet de la carte, et donc un serveur FTP.
FTP (File Transfer Protocol) est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP. Il permet, depuis un ordinateur, de copier des fichiers vers un autre ordinateur du réseau, ou encore de supprimer ou de modifier des fichiers sur cet ordinateur.

Bibliographie

[1] "Intel 4004", Internet : http://fr.wikipedia.org/wiki/Intel_4004, 26 Octobre 2013, [01 Decembre 2013].

[2] R.Litwak, "Cours d'initiation au microprocesseurs" : http://rlitwak.plil.fr/Cours_MuP/sc00a.htm, 7 Mai 2001, [01 Décembre 2013].

[3] " The apollo guidance computer ", Internet : http://fr.wikipedia.org/wiki/Apollo_Guidance_Computer, 26 Novembre, [01 Décembre 2013].

[4] "Circuit intégré ", Internet : http://fr.wikipedia.org/wiki/Circuit_int%C3%A9gr%C3%A9, 13 Octobre, [01 Decembre 2013].

[5] M.Passenaud, "Les systèmes embarqués dans l'automobile " Internet : http://synergeek.fr/les-systemes-embarques-dans-automobile, 15 janvier 2010, [05 Décembre 2013].

[6] C.Bonnet, I.Demeure, *Introduction aux systèmes temps réel*, Paris : Hermès – Lavoisier, 1999, pp14-32.

[7] J.Oudry , "Qu'est ce qu'un microcontrôleur ", Internet : http://www-igm.univ-mlv.fr/~dr/XPOSE2002/robotique/chapitres/MicrocontroleurWhat.htm, 26 juillet 2006,[08 Décembre 2013

[8] G.Blanc, *Linux embarqué,* Paris: Pearson France, 2011, pp 3-7.

[9] G.Blanc, *Linux embarqué,* Paris: Pearson France, 2011, pp 257-260.

[10] "APF28", Internet: http://www.armadeus.com/francais/produits-cartes_microprocesseur-apf28.html, 15 décembre 2013,[15 décembre 2013]

"Armadeus Project", Internet : http://www.armadeus.com/wiki/index.php?title=Main_Page, 11 juin 2013, [15 décembre 2013]

[11] "Buildroot: making Embedded Linux easy", Internet: http://buildroot.uclibc.org/, [15 décembre 2013], 15 décembre 2013

Problématique

Pourquoi utiliser une carte à microprocesseur dans un système embarqué ?

Une carte embarquée est un système autonome capable d'effectuer des taches très précises dans son environnement immédiat. L'utilisation de ce genre de carte n'a rien d'une nouveauté en soi, ceux-ci sont utilisés depuis longtemps dans des domaines comme l'aéronautique ou l'industrie. Un noyau linux a été préinstallé dans notre carte microprocesseur. Le fait d'avoir choisit linux comme système d'exploitation, se traduit par des raisons technologiques qui sont les disponibilités des sources, étant le premier critère de choix d'un OS sur un système embarqué, ainsi que sa fiabilité et sa robustesse. Et dans un second temps, linux a été choisit pour des raisons économiques car celui-ci est gratuit, simple et rapide lors de sa mise en œuvre, mais ce qui séduit particulièrement les utilisateurs de ce type de OS est son indépendance envers les fournisseurs.
Bien souvent dans l'industrie, nous avons tendances à utiliser des PC trop gros et trop lourds, avec une consommation trop élevés et pour ne pas arranger les choses leurs prix sont exorbitants. Une solution à ce problème, semble être les cartes embarquées Linux de type APF 28 de ARMADEUS dont nous avons eu l'occasion de développé lors du projet. Celle-ci présente une palette de fonctionnalités très intéressante pour les utilisateurs de systèmes embarqués, tels que mises à jour, partage de fichiers, contrôle ou configuration sécurisés par internet, et bien d'autres encore… afin d'éviter toute encombrement avec notre système embarqué, le fournisseur ARMADEUS propose de fusionner notre électronique avec la carte de développement dans le but de former une seule et unique entité, ceci pour des raisons esthétiques ou autres critères techniques et économiques. De plus, lors du développement de notre carte Linux il est possible d'avoir une assistance du fournisseur, afin de mettre en place des solutions pertinente pour un gain considérable en termes de temps.

Etat actuel

Dans notre sujet un système d'exploitation est déjà installé sur notre carte de développement. Notre objectif suivant est de mettre en place des programmes simple d'utilisation afin de manipuler les différentes entrées/sorties présentes sur la carte. Ensuite vient l'étape où nous devons réinstaller un noyau linux ainsi que tout les outils des fichiers systèmes.

Présentation du matériel

Carte à processeur :



L'APF28 est une carte à microprocesseur de taille réduite qui ne dispose pas de FPGA. Cette configuration permet d'obtenir un coût optimal pour les applications ne nécessitant pas ce composant.
Equipée d'un microprocesseur ARM9 à 454MHz, de 128 Mo RAM DDR2, de 256 Mo de FLASH, d'un port Ethernet 10/100Mbits, 2 CAN, d'un USB 2.0 High Speed et d'un USB OTG, elle est facilement intégrable dans un système embarqué grâce notamment à ses régulateurs et ses convertisseurs de niveau (USB/Ethernet).
Aucun debugger externe (BDI, JTAG) n'est requis. Nous avons utilisée avec la carte APF28, un simple câble USB lors du développement.

Carte de développement :

La carte électronique APF28_dev est une plateforme de développement idéale pour l'expérimentation et la réalisation d'applications Linux embarquées 'low cost'.
Elle permet d'accéder facilement aux fonctionnalités de l'APF28 (carte à processeur). L'ensemble des drivers permettant d'utiliser les périphériques présents sur la carte sont disponibles dans le BSP Armadeus.



Spécificités

- 1 port Ethernet
- 1 USB OTG 2.0
- 1 USB Host 2.0 (High Speed)
- 1 USB Device (console Linux)
- 1 LED utilisateur
- 1 bouton poussoir utilisateur
- 1 bouton de reset
- 1 bouton de On/Off
- Controleur de backlight pour TM035KBH02 (TFT 3"5, 320 x 240 avec dalle tactile)

Cook-book: Apf 28

Ce manuel d'utilisation est destiné à prendre en mains simplement et facilement l'Apf28 de Armadeus. Ce manuel se compose en trois parties, l'installation des outils nécessaire, une gestion simple des Entrées/Sorties et pour finir une réinstallation d'un noyau linux sur la carte.
Ce manuel a été testé sur la version de Ubuntu 13.04.

I- Installation des outils.

Pour utiliser l'Apf28 il est nécessaire d'installer certain logiciel.

1) logiciel de communication avec la carte

Pour pouvoir communiquer et contrôler la carte Armadeus APF28 nous avons besoin d'un logiciel de connections, Armadeus nous propose trois logiciel possible :

- Kermit
- MiniCom
- GTKTerm

Dans notre cas nous avons choisit GTKTerm

GTKTerm est un émulateur de terminal série qui vous permet de communiquer avec votre carte à travers d'une liaison série. Cet outil vous donne accès à la console U-Boot (en quelque sorte le "BIOS") ou au Linux de la carte.

Installation :

```
$ sudo apt-get install gtkterm
```

lancer GTKTerm:

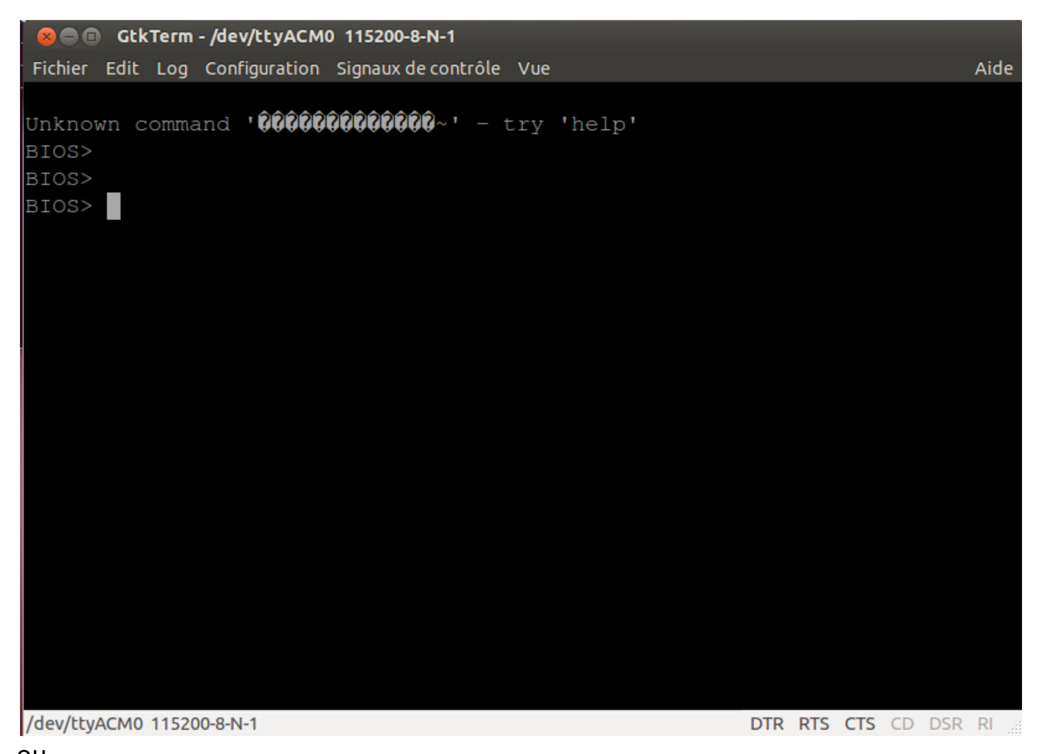
```
$ gtkterm
```

Configuration:

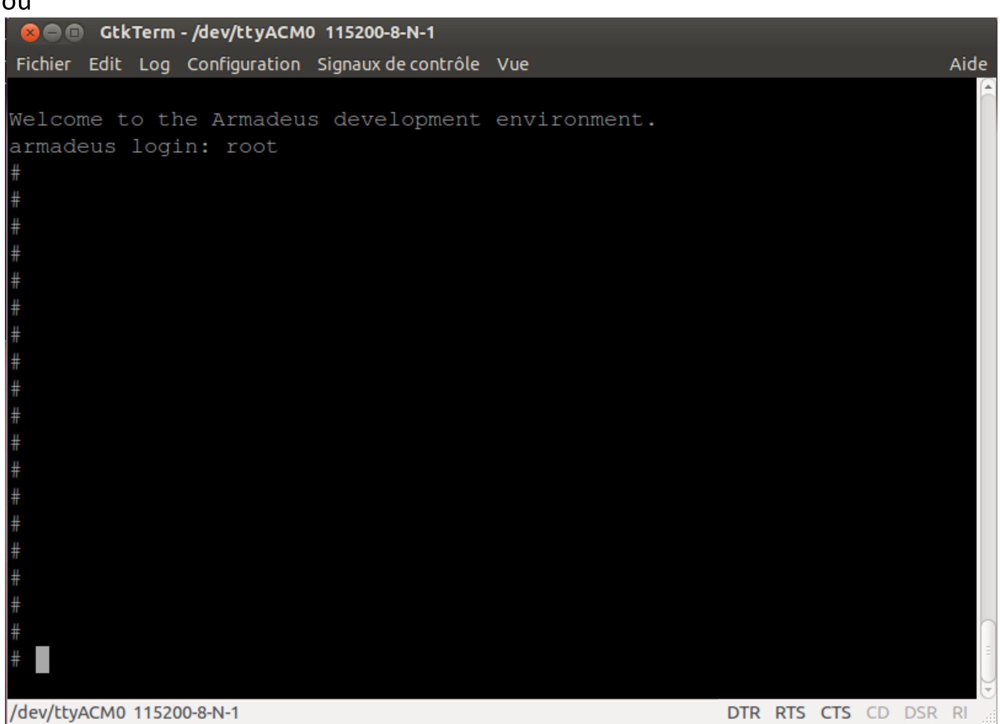
Dans le menu "Configuration" puis dans "Ports", mettez la configuration suivante:

Port: /dev/ttyACM0	Spped:115200	Parity: None
Bits: 8	Stop bit : 1	Flow control: None

Dans notre cas nous avons mis /dev/ttyACM0 dans notre port mais il se peut, suivant les machines, que la commande soit /dev/ttyS0 ou /dev/ttyUSB0.
Vous pouvez sauvegarder cette configuration, afin de simplement le chargement de la configuration à chaque fois que vous désirez communiquer avec la carte Armadeus Apf28.
Connectez votre carte en USB à votre ordinateur, lancer GTKTerm, chargez la configuration et vous devriez avoir cette fenêtre qui s'affiche:



ou



Dans le premier cas vous êtes sur la BIOS de la carte, et dans le second cas vous êtes sur le linux de la carte (si on vous demande un code d'accès mettez : "root").

Comment passez du Linux au BIOS et inversement. Lorsque vous êtes sur le BIOS pour vous connecter au Linux de la carte tapez simplement la commande suivante:

```
BIOS> boot
```

Pour aller dans le BIOS à partir de la carte il y a deux chemins possible.

- Connecter la carte à votre ordinateur en USB sans brancher la carte au secteur.
- Lorsque vous êtes sur le Linux de la carte restez appuyé le bouton "Reset" de la carte.

2) Installation du kit de développement :

Afin de pouvoir écrire des programmes pour la carte ou de changer les paramètres du système on a besoin des outils qui vont gérer la compilation croisé (ou le cross compilation). Pour cela Armadeus fournit un kit de développement. Le kit contient les logiciels suivant :

- Binutils: plusieurs utilitaires GNU utilisés pour générer les fichiers exécutables.
- Buildroot: ensemble de fichiers Makefile permettant de développer totalement un système embarqué sous Linux.
- Busybox: "couteau suisse" de Linux embarqué regroupant un bon nombre de programmes courants en un seul exécutable.
- GCC: compilateur C GNU.
- GDB: débogueur GNU.
- Rootfs: root filesystem, image du système de fichier qui sera installé sur la cible et utilisé par Linux comme point de montge de la racine ("/")
- U-Boot: bootloader, ou "BIOS" de votre système embarqué.

Pré-Installation Ubuntu:

Pour installer le kit fournit par Armadeus, il peut être nécessaire de faire cette pré-installation (Pour Ubuntu dans notre cas).

```
$ sudo apt-get install -y build-essential gcc g++ autoconf automake libtool bison flex gettext
$ sudo apt-get install -y patch subversion texinfo wget git-core
$ sudo apt-get install -y libncurses5 libncurses5-dev
$ sudo apt-get install -y zlib1g-dev liblzo2-2 liblzo2-dev
$ sudo apt-get install -y libacl1 libacl1-dev gawk cvs curl lzma
$ sudo apt-get install -y uuid-dev mercurial
$ sudo apt-get install -y python-serial python-usb
$ sudo apt-get -y install libglib2.0-dev
$ sudo apt-get -y install libnetpbm10-dev
$ sudo apt-get -y install python-xcbgen
$ sudo apt-get -y install xutils-dev
```

Vérifiez que votre système d'exploitation est en anglais, sinon il peut y avoir des erreurs de compilation :

```
...
extra/locale/locale_mmap.h:46: error: '__LOCALE_DATA_WCctype_II_LEN'
undeclared            here (not in a function)
 ...
```

pour passer votre système d'exploitation en anglais

```
$ sudo dpkg-reconfigure locales
```

Télécharger le kit à l'adresse suivante:
http://sourceforge.net/projects/armadeus/files/
Dézippez-le:

```
$ tar xjvf armadeus-5.2.tar.bz2
```

cela vous créé un dossier /armadeus/ ou /armadeus-5.2/

Lancez Buildroot:

```
$cd .../armadeus5.2/
$ make apf28_defconfig
```

Cette commande charge les configurations par défaut pour l'Apf28 et lance le menu de configuration de Buildroot.

```
Buildroot 2011.05 Configuration
Arrow keys navigate the menu.  <Enter> selects submenus --->.
Highlighted letters are hotkeys.  Pressing <Y> selectes a feature, while
<N> will exclude a feature.  Press <Esc><Esc> to exit, <?> for Help, </>
for Search.  Legend: [*] feature is selected  [ ] feature is excluded

        Target Architecture (arm)  --->
        Target Architecture Variant (arm926ej-s (arm926t))  --->
        Target ABI (EABI)  --->
        Build options  --->
        Toolchain  --->
        System configuration  --->
        Package Selection for the target  --->
        Filesystem images  --->
        Bootloaders  --->
        Kernel  --->
    ---
        Load an Alternate Configuration File
        Save an Alternate Configuration File

        <Select>    < Exit >    < Help >
```

Dans System Configuration --> [*] Armadeus Device Support

Vérifiez la configuration suivante :

```
Armadeus Device Support
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are
hotkeys.  Pressing <Y> selectes a feature, while <N> will exclude a feature.  Press
<Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] feature is selected  [ ]
feature is excluded

          --- Armadeus Device Support
          (apf51) Board name
          (imx51) CPU name
          (1)   Number of RAM chips on your board
          (256) Size of a single RAM chip (in MB)

          <Select>    < Exit >    < Help >
```

Sauvegardez et quittez Buildroot.
Si vous voulez retourner au menu de configuration de buildroot faites: dans le dossier /armadeus-5.2/

```
$ make menuconfig
```

Lancez la compilation

```
$ make
```

Si c'est la première fois que vous lancez la compilation cela peu durer un peu plus d'une heure.

3) Installation du serveur FTP:

Pour pouvoir envoyer des programmes sur la carte nous devons utiliser le port Ethernet pour cela on peux utiliser différentes technologies de communication. Dans notre cas nous avons installer un serveur FTP.

Installation d'un serveur FTP:

```
$ apt-get install vsftpd xinetd
```

Nous allons configurer le serveur pour qu'il autorise les connections anonymes.

```
$ sudo mkdir /srv/ftp
$ sudo usermod -d /srv/ftp ftp
$ sudo /etc/init.d/vsftpd restart
```

Pour mettre à dispositions des fichiers il faut les mettre dans le dossier /srv/ftp/

II Créer des programmes

1) Première manipulation "Hello World"

Avant toute chose connectez vous sur le Linux de la carte ,branchez la carte au réseau et vérifiez qu'elle peut communiquer avec votre serveur FTP, si il n'y a pas de communication, utilisez la commande ifconfig pour y remédier.

Etape 1 - Ecrire le programme en C :

Sur la machine de développement

Dans un fichier hello.c mettez le code suivant:

```
#include <stdio.h>
            #include <stdlib.h>

            int main( int argc, char *argv[])
            {
                        printf(" APF28 : Hello World\n");
                        exit(0);
            }
```

Etape 2- Compiler le ficher:

Sur la machine de développement

```
$ /..../armadeus-5.2/buildroot/output/host/usr/bin/arm-linux-gcc -o hello hello.c
```

Cette commande va créer l'exécutable hello destiné à un microprocesseur arm, si vous essayer de l'exécuter sur votre machine, il est probable que rien de se passe.

Dans le dossier /armadeus-5.2/buildroot/host/usr/bin/ il y'a un grand nombre de compilateurs différents, certain destiné à d'autres langages (C++) et à d'autres architectures.

Etape 3 -Mettre à disposition l'exécutable sur le serveur FTP :

Sur la machine de développement

copier l'exécutable dans le dossier /srv/ftp/

```
$ cp hello /srv/ftp/
```

Etape 4- Récupérer l'exécutable sur la carte:

Sur la carte Apf28

```
# ftpget  adresseIpSrvFtp hello
```

Etape 5 -Exécuter le code :

Sur la carte Apf28

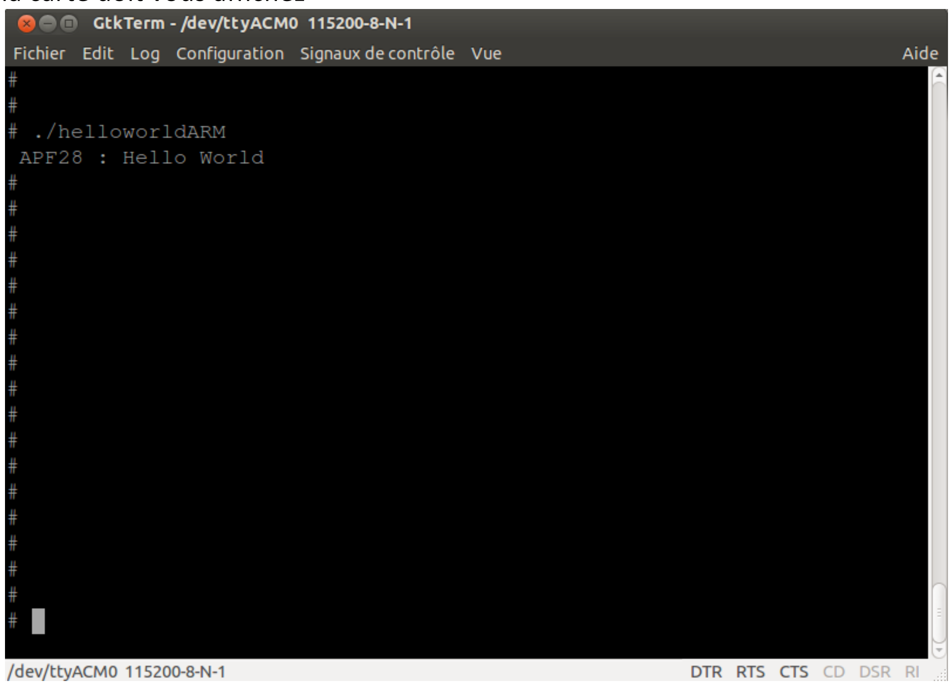
rendre les droits d'exécution à l'exécutable téléchargé.

```
# chmod +x hello
```

Et enfin lancez l'exécution

```
# ./hello
```

la carte doit vous affichez



Pour tous vos programmes respectez ces 5 étapes (6 si on compte la gestion de l'ip de la carte).

2) Gestion GPIO

On souhait un programme simple, lorsque l'on appuie sur le bouton "USER" de la carte la LED s'allume.

led_bouton.c

```
#include <stdio.h>
#include <stdlib.h>
#include "as_gpio.h"

#define LED_GPIO        21
#define SWITCH_GPIO  17

int main(int argc, char *argv[])
{
   struct as_gpio_device *led;
   struct as_gpio_device *swit;

   led = as_gpio_open(LED_GPIO);
   if (led == NULL) {
      printf("Error: can't open gpio\n");
      exit(1);
   }
         swit = as_gpio_open(SWITCH_GPIO);
         if (swit == NULL) {
      printf("Error: SWITCH can't open gpio\n");
      exit(2);
   }
         //on place la LED en sortie et le switch en entrer
         as_gpio_set_pin_direction(led, "out");
         as_gpio_set_pin_direction(swit, "in");

   while (1) {
         //si la valeur du switch est à 1 alors allumer la LED (LED=0)
         if(as_gpio_get_pin_value(swit)==1)
      {
                  as_gpio_set_pin_value(led, 0);
         }else as_gpio_set_pin_value(led, 1);
   }

   as_gpio_close(led);
   exit(0);
}
```

Les GPIO s'utilise très facilement Armadeus fournit des fichiers.c avec toutes les fonctions pour utiliser les composants.

Le fichier as_gpio.h, fournit toutes les fonctions pour ouvrir, choisir la direction de la pin et sa valeur.

Avec les fonctions : as_gpio_open() , as_gpio_get_pin_direction(), as_gpio_pin_value()

Le fichier as_gpio.c utilise des fonctions de as_helpers.c donc allez cherchez les fichiers dans le dossier /.../armadeus5-2/target/packages/as_devices/c/ : as_gpio.c, as_gpio.h, as_helpers.c, as_helpers.h et copiez les dans un dossier avec le fichier led_bouton.c .

Refaites la techniques des 5 étapes sauf que au moment de la compilation n'oubliez pas de compiler les fichiers fournit par Armadeus.

```
$ /..../armadeus-5.2/buildroot/output/host/usr/bin/arm-linux-gcc -c as_gpio.c
$ /..../armadeus-5.2/buildroot/output/host/usr/bin/arm-linux-gcc -c as_helper.c
$ /..../armadeus-5.2/buildroot/output/host/usr/bin/arm-linux-gcc -c led_bouton.c
$ /..../armadeus-5.2/buildroot/output/host/usr/bin/arm-linux-gcc -o led_bouton as_gpio.o
as_helpers.o led_bouton.o
```

ou faire un makefile qui fera ces commandes.

3) PWM

Maintenant on va gérer le PWM très simplement, en allumant la LED lorsque que le signal carré du PWM est à l'état haut.

mon_pwm.c

```
#include <stdio.h>
#include <stdlib.h>
#include "as_gpio.h"
#include "as_pwm.h"
#include "as_helpers.h"

#define LED_GPIO        21

 struct as_gpio_device *led;
 //struct as_gpio_device *swit;
 struct as_pwm_device *my_pwm;
int P;


int main(int argc, char *argv[])
{
        led = as_gpio_open(LED_GPIO);
        if (led == NULL) {
                printf("Error: can't open gpio\n");
                 exit(1);
        }

        my_pwm= as_pwm_open(0);
        if(!my_pwm)
        {
        printf("Erreur: ouverture pwm");
        exit(1);
        }

        //500HZ
        as_pwm_set_frequency(my_pwm,500);
        //50%
        as_pwm_set_duty(my_pwm,500);

         as_gpio_set_pin_direction(led, "out");

        while(1){
                if(as_pwm_get_state(my_pwm)==1)
                {
                        as_gpio_set_pin_value(led, 1);
```

```
        }
        else as_gpio_set_pin_value(led, 0);
    }
    return 0;
}
```

Avec les fonctions dans le fichier as_pwm.h (récupéré dans le dossier /.../armadeus-5.2/target/packages/as_devices/ c/) on a différentes fonctions qui nous permet de créer un signal pwm de gérer sa fréquence (as_pwm_set_frequency ()) et son ratio (as_pwm_set_duty()). Cela nous permet de simplifier le code et de pouvoir utiliser un signal PWM sans utiliser de timers.
Comme avec le code pour les gpio n'oubliez pas de faire de la compilation partiels avec les fichiers fournit.

4) Autres entrees/sorties

Comme pour le pwm ou les gpio, armadeus fournit de multiples fichiers afin de simplifier les éléments de la carte.

I2c :

Le bus i2c est un bus de données série synchrone bidirectionnel. Plusieurs équipements, maître ou esclaves, peuvent être connectés au bus. Les échanges ont toujours lieu entre un seul maître et un esclave, toujours à l'initiative du maître. Cependant, rien n'empêche à un composant de passer du statut maître à esclave et réciproquement.

Dans le fichier as_i2c.c on a toutes les fonctions de base pour utiliser l'I2c

```
as_i2c_read(...);
as_i2c_write(...);
as_i2c_read_bytes(...);
as_i2c_write_bytes(...);
as_i2c_read_msg(...);
```

Mais aussi des fonction pour gérer les esclaves sur le bus.

```
as_i2c_set_slave_addr(...);
as_i2c_get_slave_addr(...);
```

Convertisseurs Analogiques Numériques et Convertisseurs Numériques Analogique:

Des fonctions sont fournit pour utiliser les convertisseurs dans les fichiers as_adc.c et as_dac.c comme par exemple la fonction

```
as_adc_set_value_in_millivolts(...);
```

SPI:

Une liaison SPI (Serial Peripheral Interface) est un bus de données série synchrone. Les circuits communiquent suivant un shéma maître-esclaves, où le maîtres s'occupe totalement de la communication. Plusieurs esclaves peuvent coexister sur le même bus.

Dans le fichier as_spi.c nous avons des fonctions simplifiant l'utilisation de ce bus comme:

```
as_spi_set_mode(...);
as_spi_set_speed(...);
as_spi_set_bits_per_word(...);
```

III Installation de Linux sur la carte

Un système embarqué sous Linux est très similaire, à un système Linux classique, il se compose:

- d'un **bootloader**, c'est à dire les premiers bits de code exécutés lorsque le processeur démarre; il se comporte comme la combinaison du BIOS et de Grub sur les PC de bureau. Dans notre cas le **bootloader** est U-boot.
- d'un système d'exploitation, **le noyaux Linux**.
- de nombreux programmes et bibliothèque organisés en système de fichiers, le **rootfs**.

Ayant eu des problèmes avec VSFTP dans cette partie nous avons installé un autre serveur FTP: TFTP
<u>Installation de TFTP :</u>

```
$ sudo apt-get install tftpd xinetd
```

Créez le répertoire /tftpboot

```
$ sudo mkdir /tftpboot
$sudo chmode 777 /tftpboot
```

Ce répertoire contiendra tous les fichiers que le serveur exportera.
<u>Configuration du serveur:</u>
Vérifier que le fichier /etc/xinetd.d/tftp ait les configurations suivantes:

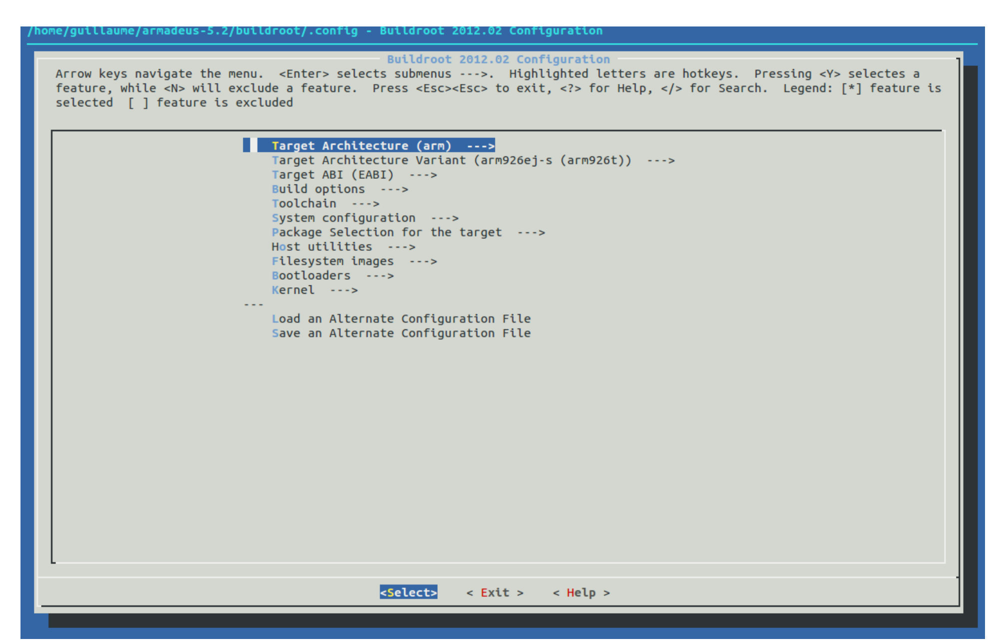
```
# default: off
# description: The tftp server serves files using the trivial file transfer
# protocol. The tftp protocol is often used to boot diskless
# workstations, download configuration files to network-aware
printers,
# and to start the installation process for some operating systems.
service tftp
{
socket_type = dgram
protocol = udp
wait = yes
user = root
server = /usr/sbin/in.tftpd
server_args = -s /tftpboot
# disable = yes
}
```

puis relancer le xinetd

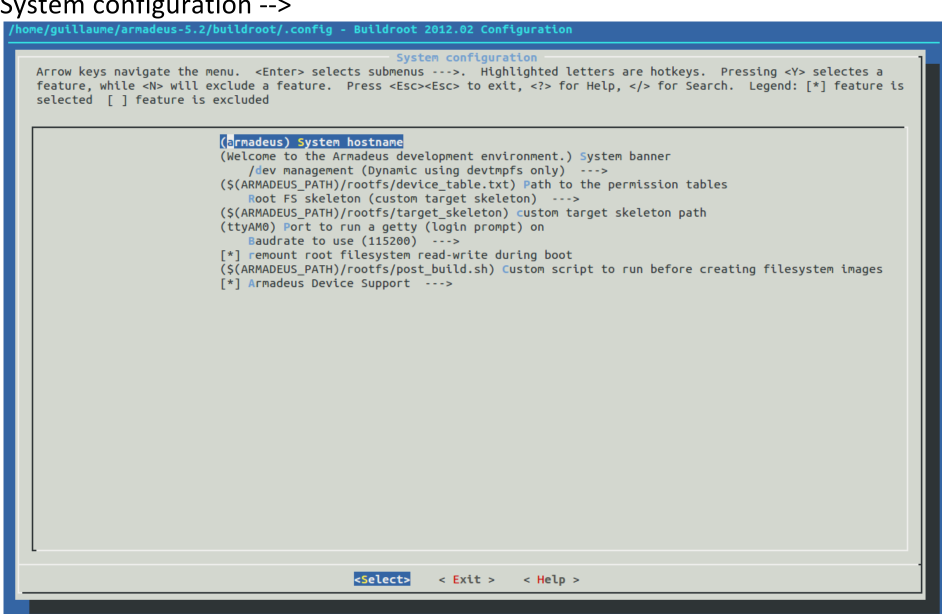
```
$ sudo killall -HUP xinetd
```

Pour créer les images du noyau du rootfs et du bootloader, il faut aller dans le dossier /../armadeus -5.2

```
$ make apf28_deconfig
```



System configuration -->



Vous pouvez configurer le rootfs, le bootloader grâce à buildroot.

```
$ make
```

Dans le dossier /armadeus-5.2/ va compiler Buildroot et créer dans le dossier /armadeus-5.2/buildroot/output/images/ les fichiers suivants:

-apf28-linux.bin : fichier image du noyau Linux
-apf28-rootf.tar : ici on a l'arborescence des fichiers géré par le rootfs
- apf28-rootfs.ubi : image du système de fichier rootfs
-apf28-rootfs.ubifs
-apf28-u-boot.bin : fichier image de U-boot -apf28-u-boot.sb

copiez ces fichiers dans le dossier /tftpboot/

Sur la carte Apf28 passez en mode BIOS et définissez une dresse ip à la carte

```
BIOS> setenv ipaddre adresseip
```

Installation du noyau Linux

Récupérez l'images sur la carte

BIOS> tftpboot *adresseServTftp* apf28-linux.bin

Après que l'image du noyau a été téléchargée en mémoire, vous pouvez la flasher à l'aide de la commande suivante:

BIOS> run flash_kernel

Installation du rootfs

Récupérez l'image du rootfs sur la carte

BIOS> tftpboot *adresseSrvTftp* apf28-rootfs.ubi

Et flashez l'image du rootfs sur la carte.

BIOS> run flash_rootfs

Maintenant, vous pouvez tester votre Linux embarqué:

BIOS> boot

Conclusion

La réalisation de ce projet a été très enrichissante. Tout d'abord, nous avons eu l'opportunité de mettre en œuvre un système embaqué sous linux, ce qui est une première en termes d'expérience. Nous avons pu revoir les bases en programmation ainsi qu'en parallélisme que nous avons acquises au cours de cette année. Nous avons également découvert des nouveaux logiciels, pour la conception et la simulation de carte embarquée.

La réalisation de ce projet nous a également donné l'opportunité de travailler sur plusieurs domaines à la fois notamment la programmation, l'électronique et l'informatique industrielle. Nous avons beaucoup appris sur la gestion de projets aussi, en effet, on n'est jamais à l'abri d'un problème, qu'il soit matériel ou logiciel. Il faut en permanence anticiper, et ne pas négliger le temps passé à la recherche de documentations. Mais la solution que nous avons apportée répond en partie à la problématique.

Le projet que l'on nous a confié n'est pas terminé, cependant nous avons des idées afin d'exploiter au mieux les fonctionnalités de notre carte. Les résultats obtenus en simulations nous permettent d'espérer une amélioration significative.

La prochaine étape du projet consistera à concevoir des programmes, mettant en œuvre des entrées et sorties dont nous n'avons pas eu le temps d'utiliser par manque de temps et de moyens. Nous avons eu l'idée de mettre en œuvre un CAN dont dispose la carte, pour mettre en œuvre l'acquisition des mesures effectuées par un capteur (capteur de température par exemple), pour ensuite afficher le résultat sur un écran LCD (Vendu par le constructeur Armadeus). Et pour cela nous avons réalisé au préalable une recherche documentaire nous permettant de mettre en œuvre ces différentes fonctions.

Grâce à l'appui de notre tuteur ainsi que la documentation mis à notre disposition, nous avons pu réaliser une partie de notre projet dans d'excellentes conditions.